

BBBBBBBBBBBB		00000000		00000000		TTTTTTTTTTTT		SSSSSSSSSS
BBBBBBBBBBBB		00000000		00000000		TTTTTTTTTTTT		SSSSSSSSSS
BBBBBBBBBBBB		00000000		00000000		TTTTTTTTTTTT		SSSSSSSSSS
BBB	BBB	000	000	000	000	TTT	SSS	
BBB	BBB	000	000	000	000	TTT	SSS	
BBB	BBB	000	000	000	000	TTT	SSS	
BBB	BBB	000	000	000	000	TTT	SSS	
BBB	BBB	000	000	000	000	TTT	SSS	
BBB	BBB	000	000	000	000	TTT	SSS	
BBB	BBB	000	000	000	000	TTT	SSS	
BBBBBBBBBBBB		000	000	000	000	TTT	SSS	
BBBBBBBBBBBB		000	000	000	000	TTT	SSS	
BBBBBBBBBBBB		000	000	000	000	TTT	SSS	
BBB	BBB	000	000	000	000	TTT	SSS	
BBB	BBB	000	000	000	000	TTT	SSS	
BBB	BBB	000	000	000	000	TTT	SSS	
BBB	BBB	000	000	000	000	TTT	SSS	
BBB	BBB	000	000	000	000	TTT	SSS	
BBB	BBB	000	000	000	000	TTT	SSS	
BBBBBBBBBBBB		00000000		00000000		TTT	SSSSSSSSSS	
BBBBBBBBBBBB		00000000		00000000		TTT	SSSSSSSSSS	
BBBBBBBBBBBB		00000000		00000000		TTT	SSSSSSSSSS	

```
DDDDDDDD  QQQQQQ  BBBB8888  TTTTTTTTTT  DDDDDDDD  RRRRRRRR  IIIIII  VV  VV  RRRRRRRR
DDDDDDDD  QQQQQQ  BBBB8888  TTTTTTTTTT  DDDDDDDD  RRRRRRRR  IIIIII  VV  VV  RRRRRRRR
DD  DD  QQ  QQ  BB  BB  TT  DD  DD  RR  RR  II  VV  VV  RR  RR
DD  DD  QQ  QQ  BB  BB  TT  DD  DD  RR  RR  II  VV  VV  RR  RR
DD  DD  QQ  QQ  BB  BB  TT  DD  DD  RR  RR  II  VV  VV  RR  RR
DD  DD  QQ  QQ  BB  BB  TT  DD  DD  RR  RR  II  VV  VV  RR  RR
DD  DD  QQ  QQ  BB  BB  TT  DD  DD  RR  RR  II  VV  VV  RR  RR
DD  DD  QQ  QQ  BB  BB  TT  DD  DD  RR  RR  II  VV  VV  RR  RR
DD  DD  QQ  QQ  BB  BB  TT  DD  DD  RR  RR  II  VV  VV  RR  RR
DD  DD  QQ  QQ  BB  BB  TT  DD  DD  RR  RR  II  VV  VV  RR  RR
DD  DD  QQ  QQ  BB  BB  TT  DD  DD  RR  RR  II  VV  VV  RR  RR
DDDDDDDD  QQQQ  QQ  BBBB8888  TTT  DDDDDDDD  RR  RR  IIIIII  VV  VV  RR  RR
DDDDDDDD  QQQQ  QQ  BBBB8888  TT  DDDDDDDD  RR  RR  IIIIII  VV  VV  RR  RR
LL  IIIIII  SSSSSSSS
LL  IIIIII  SSSSSSSS
LL  II  SS
LL  II  SS
LL  II  SS
LL  II  SSSSSS
LL  II  SSSSSS
LL  II  SS
LL  II  SS
LL  II  SS
LLLLLLLLLL  IIIIII  SSSSSSSS
LLLLLLLLLL  IIIIII  SSSSSSSS
```


(1) 73
(2) 220

DECLARATIONS
RB730:RB02/RB80 Bootstrap driver code

```

0000 1      .TITLE DQBTDRIVR - RB730:RB02/RB80 BOOT DRIVER
0000 2      .IDENT 'V04-000'
0000 3
0000 4
0000 5 *****
0000 6 *****
0000 7      *  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
0000 8      *  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
0000 9      *  ALL RIGHTS RESERVED.
0000 10
0000 11      *  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
0000 12      *  ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
0000 13      *  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
0000 14      *  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
0000 15      *  OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
0000 16      *  TRANSFERRED.
0000 17
0000 18      *  THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000 19      *  AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
0000 20      *  CORPORATION.
0000 21
0000 22      *  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0000 23      *  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
0000 24
0000 25 *****
0000 26 *****
0000 27
0000 28
0000 29      ++
0000 30      FACILITY:      BOOTS
0000 31
0000 32      ABSTRACT:
0000 33          This module contains the bootstrap device driver for
0000 34          the RB02 and RB80 disks on the RB730 controller.
0000 35
0000 36      ENVIRONMENT:  IPL 31, kernel mode, code must be PIC
0000 37
0000 38      AUTHOR:  Greg Robert, CREATION DATE:  09-Jun-1981
0000 39
0000 40      NOTE: The RB730 controller is supported by host microcode which is
0000 41          activated each time a device register is accessed, and for
0000 42          each block transferred. Because of this, registers must be
0000 43          handled in strict accordance with the RB730 software specification.
0000 44          The following special registers protocols are significant:
0000 45
0000 46          DAR --  When the DAR is loaded some drive functions
0000 47                  are initiated. Consequently the function code
0000 48                  must be loaded into the CSR (with CRDY set)
0000 49                  prior to loading the DAR for seeks and transfers.
0000 50                  For data transfers, the BAR and BCR must also
0000 51                  be loaded prior to loading the DAR.
0000 52
0000 53          MPR --  The microcode keeps an internal disk address
0000 54                  register for computing cylinder difference words
0000 55                  for RB02 seek commands. If this internal register
0000 56                  falls out of sync with the actual disk address
0000 57                  it can be reset by executing a "read header" command.

```


Note that the header must actually be read (thru
the MPR) in order to effect the reset.

```

0000 58 :
0000 59 :
0000 60 :
0000 61 : MODIFIED BY:
0000 62 :
0000 63 : V401 GRR4013 Gregory R. Robert 15-Dec-1983
0000 64 : Add recalibrate command to R80 error path to recover
0000 65 : when heads locked inside field service cylinder.
0000 66 :
0000 67 : V202 GRR2002 Gregory R. Robert 30-Nov-1981
0000 68 : Added ECC correction logic. The code was modelled after
0000 69 : the equivalent routines in DMBTDRIVR and MBBTDRIVR.
0000 70 :
0000 71 : --
0000 72 :
0000 73 : .SBTTL DECLARATIONS
0000 74 :
0000 75 : INCLUDE FILES:
0000 76 :
0000 77 :
0000 78 : $BTDEF ; Boot device types
0000 79 : $IODEF ; I/O function codes
0000 80 : $PRDEF ; Processor Register definitions
0000 81 : $RPBDEF ; RPB offsets
0000 82 : $SSDEF ; Status codes
0000 83 : $UBADEF ; UBA definitions
0000 84 : $UBIDEF ; 11/750 UBA definitions
0000 85 :
0000 86 :
0000 87 : MACROS:
0000 88 :
0000 89 :
0000 90 :
0000 91 : EQUATED SYMBOLS:
0000 92 :
0000 93 :
00000014 0000 94 MODE = 20 ; MAPPING MODE OFF ARGUMENT POINTER
0000 95 :
0000 96 :
0000 97 : RB730:/RB02/RB80 REGISTER OFFSETS FROM CSR ADDRESS
0000 98 :
0000 99 : $DEFINI RB ; START OF REGISTER DEFINITIONS
0000 100 :
0000 101 $DEF RB CS .BLKL 1 ; CONTROL STATUS REGISTER (CSR)
0000 102 _VTELD RB CS,0,<- ; START OF CSR BIT DEFINITIONS
0000 103 <DRDY,M>,- ; DRIVE READY
0000 104 <FCODE,3>,- ; FUNCTION CODE
0000 105 <2>,- ; RESERVED BITS
0000 106 <IE,M>,- ; INTERRUPT ENABLE
0000 107 <CRDY,M>,- ; CONTROLLER READY
0000 108 <DS,2>,- ; DRIVE SELECT
0000 109 <OPI,M>,- ; OPERATION INCOMPLETE
0000 110 <DCK,M>,- ; DATA CRC OR HEADER CRC OR DATA ECC
0000 111 <DLT,M>,- ; DATA LATE OR HEADER NOT FOUND
0000 112 <NXM,M>,- ; NON-EXISTENT MEMORY
0000 113 <DE,M>,- ; DRIVE ERROR
0000 114 <CE,M>,- ; COMPOSITE ERROR

```

```
0004 115 <ATN,4>,- ; DRIVE ATTENTION BITS
0004 116 <ECS,2>,- ; ECC STATUS
0004 117 <SSE1,M>,- ; SKIP SECTOR ERROR INHIBIT
0004 118 <SSE,M>,- ; SKIP SECTOR ERROR
0004 119 <IR,M>,- ; IDC INTERRUPT REQUEST
0004 120 <MTN,M>,- ; MAINTENANCE MODE
0004 121 <TYP,M>,- ; DRIVE TYPE 1=RB80, 0=RB02
0004 122 <ASS1,M>,- ; AUTOMATIC SKIP SECTOR INHIBIT
0004 123 <TOI,M>,- ; TIME OUT INHIBIT (U-DIAG'S)
0004 124 <FMT,M>,- ; R80 FORMAT CONTROL
0004 125 <,2>- ; RESERVED BITS
0004 126 > ; END CSR BIT DEFINITIONS
0004 127
0004 128 $DEF RB_BA .BLKL 1 ;BUS ADDRESS REGISTER (BAR)
0008 129
0008 130 $DEF RB_BC .BLKL 1 ;BYTE COUNT REGISTER (BCR)
000C 131
000C 132 $DEF RB_DA .BLKL 1 ;DISK ADDRESS REGISTER (DAR)
0010 133 -VIELD RB_DA,0,<- ;START OF DAR BIT DEFINITIONS
0010 134 <SEC,8>,- ; SECTOR
0010 135 <TRK,8>,- ; TRACK
0010 136 <CYL,16>- ; CYLINDER
0010 137 > ;END OF DAR BIT DEFINITIONS
0010 138
0010 139 $DEF RB_MP .BLKL 1 ;MULTIPURPOSE REGISTER (MPR)
0014 140 -VIELD RB_MP,0,<- ;RB02 STATUS WORD DEFINITIONS
0014 141 <STA,3>,- ; DRIVE STATE
0014 142 <BH,M>,- ; BRUSH HOME
0014 143 <HO,M>,- ; HEADS OUT
0014 144 <CO,M>,- ; COVER OPEN
0014 145 <HS,M>,- ; HEAD SELECT
0014 146 <,1>- ; RESERVED
0014 147 <DSE,M>,- ; DRIVE SELECT ERROR
0014 148 <VC,M>,- ; VOLUME CHECK
0014 149 <WGE,M>,- ; WRITE GATE ERROR
0014 150 <SPE,M>,- ; SPIN ERROR
0014 151 <SKTO,M>,- ; SEEK TIME OUT
0014 152 <WL,M>,- ; WRITE LOCK
0014 153 <CHE,M>,- ; CURRENT HEAD ERROR
0014 154 <WDE,M>- ; WRITE DATA ERROR
0014 155 >
0014 156 -VIELD RB_MP,0,<- ;GET STATUS COMMAND DEFINITIONS
0014 157 <MRK,M>,- ; MARK (ALWAYS 1)
0014 158 <STS,M>,- ; GET STATUS
0014 159 <,1>- ; RESERVED
0014 160 <RST,M>,- ; RESET
0014 161 >
0014 162 -VIELD RB_MP,0,<- ;RB80 STATUS WORD DEFINITIONS
0014 163 <SEC,5>,- ; CURRENT RB80 SECTOR
0014 164 <,3>- ; RESERVED
0014 165 <FLT,M>,- ; DRIVE FAULT
0014 166 <PLGV,M>,- ; PLUG VALID
0014 167 <SKE,M>,- ; SEEK ERROR
0014 168 <ONCY,M>,- ; ON CYLINDER
0014 169 <DRDY,M>,- ; DRIVE READY
0014 170 <WTP,M>,- ; WRITE PROTECT
0014 171 <,2>- ; RESERVED
```



```
0014 172 > ;END MPR BIT DEFINITIONS
0014 173
0014 174 $DEF RB_EC1 .BLKL 1 ;ECC POSITION REGISTER (EPOR)
0018 175 _VIELD RB_EC1,0,<- ;START OF EC1 BIT DEFINITIONS
0018 176 <POS,13>,- ; STARTING BIT POSITION OF ECC ERROR
0018 177 <,21>- ; RESERVED
0018 178 > ;END EC1 BIT DEFINITIONS
0018 179
0018 180 $DEF RB_EC2 .BLKL 1 ;ECC PATTERN REGISTER (EPAR)
001C 181 _VIELD RB_EC2,0,<- ;START OF EC2 BIT DEFINITIONS
001C 182 <PAT,11>,- ; PATTERN OF ECC ERROR BURST
001C 183 <,21>- ; RESERVED
001C 184 > ;END EC2 BIT DEFINITIONS
001C 185
001C 186 $DEF RB_CMD .BLKL 1 ;AUXILLARY COMMAND REGISTER
0020 187 _VIELD RB_CMD,0,<- ;START OF CMD BIT DEFINITIONS
0020 188 <INIT,32>,- ; SUBSYSTEM CLEAR <-- -1
0020 189 > ;END CMD BIT DEFINITIONS
0020 190
0020 191 $DEFEND RB ;END RB730:RB80/RB02 REGISTER DEFS
0000 192
0000 193 :
0000 194 : HARDWARE FUNCTION CODES
0000 195 :
0000 196 :
00000000 0000 197 F_NOP=0*2 ;NO OPERATION
00000006 0000 198 F_SEEK=3*2 ;SEEK CYLINDER
00000008 0000 199 F_READHEAD=4*2 ;READ HEADER
00000002 0000 200 F_WRITECHECK=1*2 ;WRITE CHECK
0000000A 0000 201 F_WRITEDATA=5*2 ;WRITE DATA
0000000C 0000 202 F_READDATA=6*2 ;READ DATA
00000004 0000 203 F_GETSTATUS=2*2 ;GET STATUS
0000 204
0000 205 :
0000 206 : OWN STORAGE:
0000 207 :
0000 208 :
0000 209 :
0000 210 :
0000 211 : Boot driver table entry
0000 212 :
0000 213 :
00C0 214 $BOOT_DRIVER CPUTYPE = PR$ SID TYP730,- ; Must be VAX 11/730
0000 215 DEVTYPE = BTD$K DQ,- ; Device type (DQ)
0000 216 SIZE = DQ_DRVSI7,- ; Driver size
0000 217 ADDR = DQ_DRIVER,- ; Driver address
0000 218 DRVRNAME = DQNAME ; Driver file name
```

```
0000 220      .SBTTL RB730:RB02/RB80 Bootstrap driver code
0000 221
0000 222      :++
0000 223
0000 224      Inputs:
0000 225
0000 226      R3      - base address of adapter's register space
0000 227      R5      - LBN FOR CURRENT PIECE OF TRANSFER
0000 228      R6      - contains 0
0000 229      R7      - address of device's CSR
0000 230      R8      - SIZE OF TRANSFER IN BYTES
0000 231      R9      - address of the RPB
0000 232      R10     - starting address of transfer (byte offset in first
0000 233                  page ORed with starting map register number)
0000 234
0000 235      FUNC(AP)- I/O operation (IOS_READBLK or IOS_WRITEBLK only)
0000 236      BUF(AP) - Buffer address
0000 237      SIZE(AP)- Size of transfer
0000 238      MODE(AP)- Transfer mapping mode
0000 239
0000 240      Implicit inputs:
0000 241
0000 242      RPB$W_UNIT      - RPB field containing boot device unit number
0000 243
0000 244      Outputs:
0000 245
0000 246      R0 - status code
0000 247
0000 248      SS$_NORMAL      - successful transfer
0000 249      SS$_CTRLERR     - fatal controller error
0000 250
0000 251      R3 - must be preserved
0000 252
0000 253      This routine destroys R1, R2, R4, R5, R6.
0000 254
0000 255
0000 256      Register usage during this routine:
0000 257
0000 258      R0 - scratch
0000 259      R1 - byte count for current transfer / scratch
0000 260      R2 - scratch
0000 261      R3 - adaptor base address
0000 262      R4 - unit number
0000 263      R5 - logical block number for current transfer
0000 264      R6 - disk address for current transfer / scratch
0000 265      R7 - device CSR
0000 266      R8 - remaining byte count
0000 267      :--
0000 268
00000004 0000 269      BUF = 4
00000008 0000 270      SIZE = 8
00000010 0000 271      FUNC = 16
0000 272
0000 273      DQ_DRIVER:
0000 274
0000 275      :
0000 276      : COMPUTE ADDRESS OF DEVICE CSR
```



```

57 0200 C3 DE 0000 277 :
0000 278 : MOVAL ^X200(R3),R7 ; COMPUTE CORRECT DEVICE CSR
0005 279 :
0005 280 :
0005 281 : POSITION AND STORE THE UNIT NUMBER IN R4 FOR GLOBAL USE
0005 282 :
54 02 08 64 A9 F0 0005 283 : CLRL R4 ; CLEAR R4
0007 284 : INSV RPB$W_UNIT(R9),#8,#2,R4 ; GET UNIT NUMBER
000D 285 :
000D 286 :
000D 287 : RESET DRIVE AND GET STATUS
000D 288 :
1C A7 01 CE 000D 289 10$: MNEGL #1,RB_CMD(R7) ; INITIALIZE SUBSYSTEM
D0 0011 290 : MOVL #RB_MP_M_STS- ; PUT GET STATUS
0012 291 : !RB_MP_M_RST- ; ... AND WITH RESET
0012 292 : !RB_MP_M_MRK,- ; ... AND MARK BIT
10 A7 0B 0012 293 : RB_MP(R7) ; ... INTO DAR
54 C9 0015 294 : BISL3 R4,- ; MERGE UNIT NUMBER
0017 295 : #F_GETSTATUS,- ; ... AND FUNCTION
67 04 0017 296 : RB_CS(R7) ; ... INTO CSR
50 10 A7 D0 0019 297 : BSBW READY ; WAIT FOR DRIVE AND CONTROLLER READY
67 04000000 8F D3 001C 298 : MOVL RB_MP(R7),R0 ; FETCH STATUS WORD
15 12 0020 299 : BITL #RB_CS_M_TYP,RB_CS(R7) ; IS THIS AN RB80?
0027 300 : BNEQ 20$ ; BRANCH IF SO
0029 301 :
0029 302 :
0029 303 : CHECK RB02 STATUS
0029 304 :
1D 50 05 00 ED 0029 305 : CMPZV #0,#5,R0,- ; TEST BITS 04:00 OF STATUS FOR
002E 306 : #RB_MP_M_HO- ; ... HEADS OUT
002E 307 : !RB_MP_M_BH- ; ... BRUSHES HOME
002E 308 : 15 ; ... SEEK LINEAR MODE (READY TO GO)
DD 12 002E 309 : BNEQ 10$ ; LOOP IF NOT READY
0030 310 :
0030 311 :
0030 312 : READ A HEADER TO MAKE SURE MICROCODE IS SYNCHRONIZED WITH CURRENT
0030 313 : DISK POSITION
0030 314 :
54 C9 0030 315 : BISL3 R4,- ; MERGE UNIT NUMBER
0032 316 : #F_READHEAD,- ; ... AND FUNCTION
67 08 0032 317 : RB_CS(R7) ; ... INTO CSR
10 A7 10 A7 D1 0034 318 : BSBW READY ; WAIT FOR DRIVE AND CONTROLLER READY
0037 319 : CMPL RB_MP(R7),RB_MP(R7) ; READ THE HEADER (UCODE DOESN'T LOOK
003C 320 : ; ...AT IT UNLESS WE ACCESS IT)
07 11 003C 321 : BRB 200$ ; CONTINUE IN COMMON
003E 322 :
003E 323 :
003E 324 :
003E 325 : CHECK RB80 STATUS
003E 326 :
1A 50 05 08 ED 003E 327 20$: CMPZV #8,#5,R0,- ; TEST BITS 08:12 OF STATUS FOR
0043 328 : #<RB_MP_M_DRDY- ; ...DRIVE READY
0043 329 : !RB_MP_M_ONCY- ; ...ON CYLINDER
0043 330 : !RB_MP_M_PLGV>- ; ...PLUG VALID
0043 331 : @-8 ; ...SHIFT TO LOW BYTE
C8 12 0043 332 : BNEQ 10$ ; IF NOT, BRANCH TO WAIT FOR IT
0045 333 :
```



```
0045 334
0045 335
0045 336 : NOW CONVERT LOGICAL TO PHYSICAL -- THE COMPUTED DISK ADDRESS HAS THE
0045 337 : FORM OF A LONG WORD WITH LOW BYTE=SECTOR, NEXT BYTE=TRACK, AND HIGH
0045 338 : WORD = CYLINDER.
0045 339
0045 340 : I) CYLINDER = LBN / BLOCKS_PER_CYLINDER
0045 341 : II) TRACK = REMAINDER(I) / BLOCKS_PER_TRACK
0045 342 : III) RB02_SECTOR = REMAINDER(II) * 2
0045 343 : IV) RB80_SECTOR = REMAINDER(II)
0045 344
0045 345
67 04000000 51 D4 0045 346 200$: CLRL R1 : CLEAR HIGH PART OF DIVIDEND
56 D4 0047 347 CLRL R6 : CLEAR HIGH PART OF DIVIDEND
8F D3 0049 348 BITL #RB_CS_M_TYP,RB_CS(R7) : IS THIS AN RB80?
13 12 0050 349 BNEQ 220$ : BRANCH IF SO
0052 350
0052 351
0052 352 : COMPUTE RB02 DISK ADDRESS
0052 353
50 52 55 28 7B 0052 354 EDIV #40/2*2,R5,R2,R0 : R2 = DESIRED CYL, R0 = REMAINING BLKS
56 50 50 14 7B 0057 355 EDIV #40/2,R0,R0,R6 : R0 = DESIRED TRK, R6 = REMAINING BLKS
56 56 02 C4 005C 356 MULL #2,R6 : 2 SECTORS PER BLOCK
51 28 56 C3 005F 357 SUBL3 R6,#40,R1 : R1 = 256 BYTE SECTORS LEFT ON SURFACE
15 11 0063 358 BRB 230$ : CONTINUE IN COMMON
0065 359
0065 360 : COMPUTE RB80 DISK ADDRESS INSTEAD
0065 361
0065 362
50 52 55 000001B2 8F 7B 0065 363 220$: EDIV #31*14,R5,R2,R0 : R2 = DESIRED CYL, R0 = REMAINING BLKS
56 50 50 1F 7B 006E 364 EDIV #31,R0,R0,R6 : R0 = DESIRED TRK, R6 = REMAINING BLKS
51 1F 56 C3 0073 365 SUBL3 R6,#31,R1 : R1 = 512 BYTE SECTORS LEFT ON SURFACE
51 02 C4 0077 366 MULL #2,R1 : R1 = 256 BYTE SECTORS LEFT ON SURFACE
007A 367
007A 368
007A 369 : IF FINAL TRANSFER USE REMAINING BYTE COUNT, ELSE USE REMMAINDER OF TRACK
007A 370
51 00000100 8F C4 007A 371 230$: MULL #256,R1 : R1 = BYTES LEFT ON SURFACE
51 58 D1 0081 372 CMPL R8,R1 : ARE ADDITIONAL TRANSFERS REQUIRED?
03 1A 0084 373 BGTRU 240$ : BRANCH IF ANSWER YES
51 58 D0 0086 374 MOVL R8,R1 : SET BYTE COUNT FOR FINAL TRANSFER
0089 375
0089 376
0089 377 : FORM FULL DISK ADDRESS (CYL, TRK, SEC)
0089 378
56 10 10 52 F0 0089 379 240$: INSV R2,#16,#16,R6 : MOVE CYLINDER INTO HIGH WORD
56 08 08 50 F0 008E 380 INSV R0,#8,#8,R6 : MOVE TRACK INTO SECOND BYTE
0093 381
0093 382
0093 383 : PERFORM SEEK -- NOTE: FOR RB730 SEEKS AND TRANSFERS, THE COMMAND
0093 384 : MUST BE LOADED INTO THE CSR WITH CONTROLLER READY BIT SET, BEFORE
0093 385 : WRITING TO THE DISK ADDRESS REGISTER !!!
0093 386
54 C9 0093 387 BISL3 R4,- : MERGE UNIT NUMBER
0095 388 #F,SEEK- : ... FUNCTION AND
0095 389 !RB_CS_M_CRDY,- : ... SUPPRESS EXECUTION
67 00000086 8F 0095 390 RB_CS(R7) : ... INTO CSR
```



```
67 0C A7 56 D0 009B 391      MOVL    R6,RB_DA(R7)      ; LOAD DISK ADDRESS IN DAR
    00000080 8F CA 009F 392      BICL    #RB_CS_M_CRDY,RB_CS(R7) ; INITIATE THE FUNCTION
    00EE 30 00A6 393      BSBW    READY      ; WAIT FOR CONTROLLER AND DRIVE READY
    03 50 0F E1 00A9 394      BBC     #RB_CS_V_CE,R0,300$    ; BRANCH IF NO ERRORS
    00AF 31 00AD 395      BRW     900$      ; EXIT
    00B0 396
    00B0 397
    00B0 398
    00B0 399 : SEEK IS COMPLETE -- EXECUTE TRANSFER FUNCTION
    00B0 400
52 8C 8F 9A 00B0 401 300$: MOVZBL #F_READDATA-      ; ASSUME READ FUNCTION
    00B4 402      !RB_CS_M_CRDY,R2      ; ... WITH CONTROLLER READY
20 10 AC D1 00B4 403      CML     FUNC(AP),#10$_WRITELBLK ; IS IT A WRITE FUNCTION?
    04 12 00B8 404      BNEQ    350$      ; BRANCH IF NOT
52 8A 8F 9A 00BA 405      MOVZBL #F_WRITEDATA-      ; SET WRITE FUNCTION CODE
    00BE 406      !RB_CS_M_CRDY,R2      ; ... WITH CONTROLLER READY
    00BE 407
    00BE 408
    00BE 409 : NOTE: THE DEVICE REGISTERS MUST BE LOADED IN THE PRESCRIBED ORDER!
    00BE 410      1) FUNCTION CODE LOADED INTO CSR WITH CRDY SET
    00BE 411      2) BYTE COUNT AND MEMORY ADDRESS (THESE TWO ARE REVERSABLE)
    00BE 412      3) DISK ADDRESS (FIRST HARDWARE SILO LOADED AT THIS TIME)
    00BE 413      4) CRDY CLEARED (SECOND SILO LOADED, XFER BEGINS)
    00BE 414
67 52 54 C9 00BE 415 350$: BISL3 R4,R2,-      ; MERGE UNIT NUMBER AND FUNCTION
    00C2 416      RB_CS(R7)      ; ...INTO CSR
    00C2 417      MNEGL    R1,RB_BC(R7)      ; SET NEG TRANSFER BYTE COUNT
    04 A7 5A D0 00C6 418      MOVL    R10,RB_BA(R7)      ; SET BUFFER ADDRESS
    0C A7 56 D0 00CA 419      MOVL    R6,RB_DA(R7)      ; SET DESIRED DISK ADDRESS
67 00000080 8F CA 00CE 420      BICL    #RB_CS_M_CRDY,RB_CS(R7) ; INITIATE THE FUNCTION
    00BF 30 00D5 421      BSBW    READY      ; WAIT FOR CONTROLLER AND DRIVE READY
    5A 04 A7 D0 00D8 422      MOVL    RB_BA(R7),R10      ; UPDATE BUFFER ADDRESS
    51 08 A7 C0 00DC 423      ADDL    RB_BC(R7),R1      ; COMPUTE ACTUAL BYTES TRANSFERRED
    58 51 C2 00E0 424      SUBL    R1,R8      ; UPDATE BYTES LEFT TO TRANSFER
52 51 F7 8F 78 00E3 425      ASHL    #-9,R1,R2      ; CONVERT TO PAGE COUNT
    55 52 C0 00E8 426      ADDL    R2,R5      ; UPDATE LOGICAL BLOCK NUMBER
    58 50 0F E1 00EB 427      BBC     #RB_CS_V_CE,R0,500$    ; BRANCH IF NO ERRORS
    00EF 428
    00EF 429
    00EF 430 : CHECK FOR ECC ERROR -- TO BE ELIGIBLE FOR CORRECTION THE FOLLOWING MUST APPLY
    00EF 431      THE OPERATION MUST BE A READ DATA
    00EF 432      THE ERROR MUST BE A DATA CHECK
    00EF 433      THE ERROR MUST BE A CORRECTABLE ECC ERROR
    00EF 434      THE TRANSFER MODE MATCHES THE MAP-ENABLED POSITION, I.E.,
    00EF 435      TRANSFER IS VIRTUAL, AND MAPPING IS ENABLED, OR V.V.
    00EF 436      SOME DATA WAS TRANSFERRED
    00EF 437      NO OTHER ERRORS OCCURED
    00EF 438
    00EF 439
    00EF 440
    06 50 01 ED 00EF 441      CMPZV    #RB_CS_V_FCODE,-      ; WAS THIS A READ DATA OPERATION?
    00F1 442      #RB_CS_S_FCODE,R0,-      ; ...
    00F4 443      #<F_READDATA @ -1>      ; ...
    65 50 69 12 00F4 444      BNEQ    900$      ; BRANCH IF NOT
    00F6 445      BBC     #RB_CS_V_TYP,R0,900$    ; BRANCH IF RB02
    B3 00FA 446      BITW    #RB_CS_M_DE-      ; DRIVE ERROR
    00FB 447      !RB_CS_M_NXM-      ; ...OR NON EXISTENT MEMORY
```



```
00FB 448 !RB_CS_M_DLT- ;...OR DATA LATE
00FB 449 !RB_CS_M_OPI,- ;...OR OPERATION INCOMPLETE (HDR CRC)
50 7400 8F 12 00FB 450 RO ;
5A 50 0B E1 0101 451 BNEQ 900$ ;BRANCH IF SO
14 02 02 ED 0105 452 BBC #RB_CS_V_DCK,R0,900$ ;BRANCH IF NOT A DATACHECK
03 50 0107 453 CMPZV #RB_CS_V_ECS,- ;COMPARE ECC STATUS BITS (START)
0108 454 #RB_CS_S_ECS,- ;... (SIZE)
010A 455 RO,- ;... (FROM)
010A 456 #^B11 ;...TO BINARY 11 (BOTH SET)
50 53 12 010A 457 BNEQ 900$ ;BRANCH IF NOT CORRECTABLE
14 50 38 DB 010C 458 MFPR #PR$ MAPEN,R0 ;GET MAP ENABLE STATE
AC 50 D1 010F 459 CMPL R0,MODE(AP) ;SAME AS I/O MODE
4A 12 0113 460 BNEQ 900$ ;NO, CANT DO SIMPLE ECC
0115 461 ;
0115 462 ;
0115 463 ; Appears to be a correctable data check. Attempt the correction.
0115 464 ; Register usage: R0=Position, R1=Width, R2=Buffer offset to error sector
0115 465 ;
50 14 A7 D0 0115 466 MOVL RB_EC1(R7),R0 ; FETCH ECC POSITION REGISTER
50 50 D7 0119 467 DECL R0 ; MAKE POSITION 1 ORIGIN
51 0200 C8 9E 011B 468 MOVAB 512(R8),R1 ; BYTES REMAINING PLUS ERROR SECTOR
52 08 AC 51 C3 0120 469 SUBL3 R1,SIZE(AP),R2 ; BUFFER OFFSET TO ERROR SECTOR
51 08 C4 0125 470 MULL #8,R1 ; CONVERT BYTE COUNT TO BIT COUNT
51 50 C2 0128 471 SUBL R0,R1 ; COMPUTE CORRECTION FIELD WIDTH
0B 1A 15 012B 472 BLEQ 500$ ; BR IF NO CORRECTION NEEDED
0B 51 D1 012D 473 CMPL R1,#RB_EC2_S_PAT ; MINIMUM OF 11 AND BUFFER REMAINING
03 15 0130 474 BLEQ 480$ ; KEEP MINIMUM VALUE
51 0B D0 0132 475 MOVL #RB_EC2_S_PAT,R1 ; USE MAXIMUM FIELD PATTERN WIDTH
7E 04 BC42 51 50 EF 0135 476 480$: EXTZV R0,R1,@BUF(AP)[R2],-(SP) ; GET FIELD TO BE CORRECTED
04 BC42 51 6E 18 A7 CC 013C 477 XORL RB_EC2(R7),(SP) ; APPLY CORRECTION CODE
50 8E F0 0140 478 INSV (SP)+,R0,R1,@BUF(AP)[R2] ; AND RESTORE IN BUFFER
0147 479 ;
58 D5 0147 480 500$: TSTL R8 ; MORE TO TRANSFER?
03 15 0149 481 BLEQ 600$ ; BRANCH IF DONE
FEF7 31 014B 482 BRW 200$ ; CONTINUE TRANSFER
014E 483 ;
014E 484 ;
014E 485 ; TRANSFER COMPLETE - RETURN
014E 486 ;
51 08 AC 3C 014E 487 600$: MOVZWL SIZE(AP),R1 ; SET TOTAL BYTES TRANSFERRED
07 12 0152 488 BNEQ 610$ ; BRANCH IF ORIGINAL SIZE WAS TRANSFERRED
51 00008000 8F D0 0154 489 MOVL #^X8000,R1 ; ELSE SIZE WAS FORCED TO 64K
50 01 3C 015B 490 610$: MOVZWL #SS$_NORMAL,R0 ; SET COMPLETION CODE
05 015E 491 RSB ; AND RETURN
015F 492 ;
015F 493 ;
015F 494 ; RETRY ERROR
015F 495 ;
015F 496 ;
67 04000000 8F D3 015F 497 900$: BITL #RB_CS_M_TYP,RB_CS(R7) ; IS THIS AN RB80?
16 13 0166 498 BEQL 905$ ; BRANCH IF NOT
0168 499 ;
0168 500 ; Do a recalibrate to recover from possibility that heads are locked
0168 501 ; inside field service cylinder -- unit number and function code must
0168 502 ; be loaded before disk address
0168 503 ;
54 C9 0168 504 BISL3 R4,- ; MERGE UNIT NUMBER
```



```

        016A 505      #F SEEK-      : ... FUNCTION AND
        016A 506      !RB_CS_M_CRDY,- : ... SUPPRESS EXECUTION
67      00000086 8F      016A 507      RB_CS(R7) : ... INTO CSR
        0C A7 01      CE 0170 508      #1,RB_DA(R7) : LOAD -1 (RECALIBRATE) IN DAR
67      00000080 8F      CA 0174 509      #RB_CS_M_CRDY,RB_CS(R7) : INITIATE THE FUNCTION
        0019 30 017B 510      BSBW READY : WAIT FOR CONTROLLER AND DRIVE READY
        58 08 AC 3C 017E 511 905$: MOVZWL SIZE(AP),R8 : RESTORE BYTE SIZE OF TRANSFER IN R8
        07 12 0182 512      BNEQ 910$ : BRANCH IF SIZE WAS LEGAL
5A      58 00001F40 8F      DO 0184 513      MOVL #8000,R8 : ELSE FORCE TO 64K SIZE
04      AC 09 00      EF 018B 514 910$: EXTZV #0,#9,BUF(AP),R10 : RESTORE BYTE OFFSET IN R10
        50 0054 8F      3C 0191 515      MOVZWL #SS$_CTRLERR,R0 : SET FATAL CONTROLLER ERROR
        05 0196 516      RSB : AND ATTEMPT RETRY
        0197 517
        0197 518
        0197 519 :
        0197 520 : SUBROUTINE TO WAIT FOR CONTROLLER AND DRIVE READY OR ERROR
        0197 521 :
        0197 522
        0197 523 READY:
        0197 524      MOVL RB_CS(R7),R0 : FETCH CSR
        08 50 67      DO 0197 524      BBS #RB_CS_V_CE,R0,10$ : EXIT IF ERROR
        0F      E0 019A 525      BBC #RB_CS_V_CRDY,R0,READY : LOOP UNTIL CONTROLLER READY
        F5 50 07      E1 019E 526      BBC #RB_CS_V_DRDY,R0,READY : LOOP UNTIL DRIVE READY
        F1 50 00      E1 01A2 527      RSB :
        05 01A6 528 10$:
        01A7 529
58 45 2E 52 45 56 49 52 44 51 44 00' 01A7 530 DQNAME: .ASCIC /DQDRIVER.EXE/ : Driver file name
        45 01B3
        0C 01A7
        01B4 531
        000001B4 01B4 532 DQ_DRVSIZ=-DQ_DRIVER
        01B4 533
        01B4 534      .END
```

DQBTDRIVR
Symbol table

- RB730:RB02/RB80 BOOT DRIVER

F 10

15-SEP-1984 23:49:35 VAX/VMS Macro V04-00
4-SEP-1984 23:04:13 [BOOTS.SRC]DQBTDRIVR.MAR;1

Page 11
(2)

\$TABLE	= 00000000	R	02	OPS_CVTGB	= 000048FD
BTDSK_DQ	= 00000003			OPS_CVTGF	= 000033FD
BUF	= 00000004			OPS_CVTGH	= 000056FD
DQNAME	= 000001A7	R	03	OPS_CVTGL	= 00004AFD
DQ_DRIVER	= 00000000	R	03	OPS_CVTGW	= 000049FD
DQ_DRVSIZ	= 000001B4			OPS_CVTHB	= 000068FD
FUNC	= 00000010			OPS_CVTHD	= 0000F7FD
F_GETSTATUS	= 00000004			OPS_CVTHF	= 0000F6FD
F_NOP	= 00000000			OPS_CVTHG	= 000076FD
F_READDATA	= 0000000C			OPS_CVTHL	= 00006AFD
F_READHEAD	= 00000008			OPS_CVTHW	= 000069FD
F_SEEK	= 00000006			OPS_CVTLD	= 0000006E
F_WRITECHECK	= 00000002			OPS_CVTLF	= 0000004E
F_WRITEDATA	= 0000000A			OPS_CVTLG	= 00004EFD
IOS_WRITEBLK	= 00000020			OPS_CVTLH	= 00006EFD
MODE	= 00000014			OPS_CVTLP	= 000000F9
OPS_ACB	= 0000006F			OPS_CVTPL	= 00000036
OPS_ACBF	= 0000004F			OPS_CVTPL	= 00000008
OPS_ACBG	= 00004FFD			OPS_CVTPT	= 00000024
OPS_ACBH	= 00006FFD			OPS_CVTRDL	= 0000006B
OPS_ADDD2	= 00000060			OPS_CVTRFL	= 0000004B
OPS_ADDD3	= 00000061			OPS_CVTRGL	= 00004BFD
OPS_ADDF2	= 00000040			OPS_CVTRHL	= 00006BFD
OPS_ADDF3	= 00000041			OPS_CVTSP	= 00000009
OPS_ADDG2	= 000040FD			OPS_CVTTP	= 00000026
OPS_ADDG3	= 000041FD			OPS_CVTWD	= 0000006D
OPS_ADDH2	= 000060FD			OPS_CVTWF	= 0000004D
OPS_ADDH3	= 000061FD			OPS_CVTWG	= 00004DFD
OPS_ADDP4	= 00000020			OPS_CVTWH	= 00006DFD
OPS_ADDP6	= 00000021			OPS_DIVD2	= 00000066
OPS_ASHP	= 000000F8			OPS_DIVD3	= 00000067
OPS_CLRD	= 0000007C			OPS_DIVF2	= 00000046
OPS_CLRF	= 000000D4			OPS_DIVF3	= 00000047
OPS_CLRG	= 0000007C			OPS_DIVG2	= 000046FD
OPS_CLRH	= 00007CFD			OPS_DIVG3	= 000047FD
OPS_CMPD	= 00000071			OPS_DIVH2	= 000066FD
OPS_CMPF	= 00000051			OPS_DIVH3	= 000067FD
OPS_CMPG	= 000051FD			OPS_DIVP	= 00000027
OPS_CMPH	= 000071FD			OPS_EDITPC	= 00000038
OPS_CMPP3	= 00000035			OPS_EMODD	= 00000074
OPS_CMPP4	= 00000037			OPS_EMODF	= 00000054
OPS_CRC	= 0000000B			OPS_EMODG	= 000054FD
OPS_CVTBD	= 0000006C			OPS_EMODH	= 000074FD
OPS_CVTBF	= 0000004C			OPS_MATCHC	= 00000039
OPS_CVTBG	= 00004CFD			OPS_MNEGD	= 00000072
OPS_CVTBH	= 00006CFD			OPS_MNEGF	= 00000052
OPS_CVTDB	= 00000068			OPS_MNEGG	= 000052FD
OPS_CVTDF	= 00000076			OPS_MNEGH	= 000072FD
OPS_CVTDH	= 000032FD			OPS_MOVD	= 00000070
OPS_CVTDL	= 0000006A			OPS_MOVF	= 00000050
OPS_CVTDW	= 00000069			OPS_MOVEG	= 000050FD
OPS_CVTFB	= 00000048			OPS_MOVH	= 000070FD
OPS_CVTFD	= 00000056			OPS_MOVP	= 00000034
OPS_CVTFG	= 000099FD			OPS_MOVTC	= 0000002E
OPS_CVTFH	= 000098FD			OPS_MOVTUC	= 0000002F
OPS_CVTFL	= 0000004A			OPS_MULD2	= 00000064
OPS_CVTFW	= 00000049			OPS_MULD3	= 00000065

DQBTDRIVR
Symbol table

- RB730:RB02/RB80 BOOT DRIVER

G 10

15-SEP-1984 23:49:35 VAX/VMS Macro V04-00
4-SEP-1984 23:04:13 [BOOTS.SRC]DQBTDRIVR.MAR;1

Page 12
(2)

OP\$ _MULF2 = 00000044
OP\$ _MULF3 = 00000045
OP\$ _MULG2 = 000044FD
OP\$ _MULG3 = 000045FD
OP\$ _MULH2 = 000064FD
OP\$ _MULH3 = 000065FD
OP\$ _MULP = 00000025
OP\$ _POLYD = 00000075
OP\$ _POLYF = 00000055
OP\$ _POLYG = 000055FD
OP\$ _POLYH = 000075FD
OP\$ _SCANC = 0000002A
OP\$ _SKPC = 0000003B
OP\$ _SPANC = 0000002B
OP\$ _SUBD2 = 00000062
OP\$ _SUBD3 = 00000063
OP\$ _SUBF2 = 00000042
OP\$ _SUBF3 = 00000043
OP\$ _SUBG2 = 000042FD
OP\$ _SUBG3 = 0000-3FD
OP\$ _SUBH2 = 000062FD
OP\$ _SUBH3 = 000063FD
OP\$ _SUBP4 = 00000022
OP\$ _SUBP6 = 00000023
OP\$ _TSTD = 00000073
OP\$ _TSTF = 00000053
OP\$ _TSTG = 000053FD
OP\$ _TSTH = 000073FD
PR\$ _MAPEN = 00000038
PR\$ _SID_TYP730 = 00000003
RB _BA = 00000004
RB _BC = 00000008
RB _CMD = 0000001C
RB _CS = 00000000
RB _CS _M _CRDY = 00000080
RB _CS _M _DE = 00004000
RB _CS _M _DLT = 00001000
RB _CS _M _NXM = 00002000
RB _CS _M _OPI = 00000400
RB _CS _M _TYP = 04000000
RB _CS _S _ECS = 00000002
RB _CS _S _FCODE = 00000003
RB _CS _V _CE = 0000000F
RB _CS _V _CRDY = 00000007
RB _CS _V _DCK = 0000000B
RB _CS _V _DRDY = 00000000
RB _CS _V _ECS = 00000014
RB _CS _V _FCODE = 00000001
RB _CS _V _TYP = 0000001A
RB _DA = 0000000C
RB _EC1 = 00000014
RB _EC2 = 00000018
RB _EC2 _S _PAT = 0000000B
RB _MP = 00000010
RB _MP _M _BH = 00000008
RB _MP _M _DRDY = 00001000
RB _MP _M _HO = 00000010

RB _MP _M _MRK = 00000001
RB _MP _M _ONCY = 00000800
RB _MP _M _PLGV = 00000200
RB _MP _M _RST = 00000008
RB _MP _M _STS = 00000002
READY = 00000197 R 03
RPBSW _UNIT = 00000064
SIZ... = 00000020
SIZE = 00000008
SS\$ _CTRLERR = 00000054
SS\$ _NORMAL = 00000001

! Psect synopsis !

PSECT name	Allocation	PSECT No.	Attributes
ABS	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
\$ABSS	00000020 (32.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
BOOTDRIVR_4	00000028 (40.)	02 (2.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
BOOTDRIVR_2	000001B4 (436.)	03 (3.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE

! Performance indicators !

Phase	Page faults	CPU Time	Elapsed Time
Initialization	36	00:00:00.08	00:00:00.55
Command processing	134	00:00:00.78	00:00:02.47
Pass 1	586	00:00:21.50	00:00:43.83
Symbol table sort	0	00:00:02.49	00:00:05.29
Pass 2	113	00:00:05.19	00:00:10.61
Symbol table output	21	00:00:00.18	00:00:00.24
Psect synopsis output	2	00:00:00.03	00:00:00.03
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	894	00:00:30.27	00:01:03.03

The working set limit was 2000 pages.
103299 bytes (202 pages) of virtual memory were used to buffer the intermediate code.
There were 90 pages of symbol table space allocated to hold 1596 non-local and 16 local symbols.
3286 source lines were read in Pass 1, producing 14 object records in Pass 2.
145 pages of virtual memory were used to define 143 macros.

! Macro library statistics !

Macro library name	Macros defined
\$255\$DUA28:[BOOTS.OBJ]BOOTS.MLB;1	1
\$255\$DUA28:[SYS.OBJ]LIB.MLB;1	4
\$255\$DUA28:[SYSLIB]STARLET.MLB;2	8
TOTALS (all libraries)	13

1611 GETS were required to define 13 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS\$:DQBTDRIVR/OBJ=OBJ\$:DQBTDRIVR MASD\$:[EMULAT.SRC]MISSING/UPDATE=(MASD\$:[EMULAT.ENH]MISSING)+MASD\$:[BOOTS.SRC]DQBTDRIVR/

0038 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY